REMARKS

Applicant submits this Amendment with a Request for Continued Examination.

Claims 1, 2, 7 and 8 were rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 5,710,896 to Seidl (hereinafter "Seidl").

Claims 3-5 and 9-11 were rejected under 35 U.S.C. 103(a) as being obvious over Seidl in view of U.S. Patent No. 5,437,029 to Sinha (hereinafter "Sinha").

Applicant submits that the prior claims were allowable over the cited references. However, in order to expedite prosecution, Applicant has amended various of the independent claims to clarify the subject matter being claimed.

Claim 1 recites:

> 1. (Currently Amended) A storage medium containing program instructions readable by a computer for detecting and resolving circular flow paths disposed within a flow diagram representing the logical operation of a corresponding application program, the flow diagram formed by interconnecting a plurality of symbolic representations of program objects, the program objects configured to execute associated functions in response to corresponding triggering events, the readable program instructions comprising program instructions for:
>
> establishing a busy indicator at a given program object, the busy indicator signifying whether the given program object is currently executing its associated function during execution of the flow diagram;
>
> in response to the occurrence of the given program object's triggering event during execution of the flow diagram, testing the respective busy indicator;
>
> if the busy indicator signifies that the given program object is currently executing, blocking the given program object from re-executing in response to the triggering event;
>
> if the busy indicator signifies that the given program object is not currently executing, permitting the given program object to execute in response to the triggering event.

Seidl discloses a program development system to be used by a person developing application programs, the system presenting graphical icons which the person connects by lines using a computer mouse. A more detailed description of Seidl is given at pages 12-13 of the Applicants' amendment filed on June 30, 2004.

The Applicants respectfully urge that Seidl does not teach or otherwise suggest ***establishing a busy indicator at a given program object, the busy indicator signifying whether the given program object is currently executing its associated function during execution of the flow diagram***, as recited in Applicants' claim 1.

Applicants claim a run time ***busy indicator at a given program object*** which signifies ***whether the given program object is <u>currently executing</u> its associated function <u>during execution of the flow diagram</u>***. In sharp contrast, Seidl simply discloses a programming aid for a developer as he is developing an application program. Thus, the teachings in Seidl are applicable to program objects at application-development time (i.e., when the application program is being developed by graphically interconnecting the program objects), whereas the Applicants' claimed busy indicator is utilized at run time (i.e., when the application program's program objects are actually executed).

Further, the programming aid described in Seidl is not a "busy indicator" as recited in present claim 1. The programming aid described in Seidl relates to preventing an input connection of a program object from feeding back on itself, i.e., feeding back on another input of the same program object. The programming aid described in Seidl has nothing to do with whether the program object is busy executing its function, and is further unrelated to whether the program object is busy executing its function during execution of the flow diagram.

Accordingly, Seidl is legally precluded from anticipating the Applicants' claim 1 under 35 U.S.C. §102(b) because of its complete absence of ***establishing a busy indicator at a given program object, the busy indicator signifying whether the given program object is currently executing its associated function during execution of the flow diagram***, as claimed. As such, Applicants' independent claim 1 is believed to be patentable over Seidl.

The Applicants respectfully submit that independent claims 7, 13-15 and 21-23 contain the same patentable subject matter as independent claim 1, and are therefore allowable for the same reasons. The Applicants respectfully submit that claims 2-6, 8-12 and 16-20 depend on allowable claims 1, 7 and 15 and are thus also allowable.

The Examiner's arguments are next addressed.

At page 2 of the final Office action dated December 3, 2004, the Examiner states that Seidl discloses a "busy indicator signifying whether the given program object is currently executing," citing Seidl at Col. 10, lines 60-65 and at Col. 11, lines 11-14. Later, at page 5 of the Office action, the Examiner cites to Col. 14, lines 17-25 in Seidl as allegedly disclosing the Applicants' claimed busy indicator.

The Applicants respectfully submit that the Examiner's cited passages in Seidl, whether taken alone or in combination, fail to teach or suggest ***establishing a busy indicator at a given program object, the busy indicator signifying whether the given program object is currently executing its associated function during execution of the flow diagram***, as claimed.

Seidl at Col. 10, lines 59-65 states:

> Second, as long as the cursor stays within the snapping radius, the connection stays snapped to the speaker port. This gives important feedback that something has actually happened during the drag. Without this feedback, some trackers would require a user to "just let go" over a port. Then, the tracker would have to do all their hit-detection and type-checking after the fact.

The above-noted passage in Seidl describes how an application program developer may graphically attach a line to a port of a speaker icon while developing an application program containing the icon. *See* Seidl, Figs. 6-7. The developer drags her computer mouse until the cursor is within a "snapping radius" of the port. Within this radius, the line stays "snapped" to the port, thereby providing the developer visual feedback that a connection has been made.

Because Col. 10, lines 59-65 of Seidl describes connecting a line to a graphical icon while developing an application program, this passage is not applicable to the Applicants' claimed <u>run time</u> busy indicator that signifies whether a given program object is currently *executing* its associated function <u>during execution of the flow diagram</u>. Indeed, the speaker icon in Seidl does not execute any associated functions while the developer connects a line to the icon, since the application program containing the icon is still in the process of being developed.

Seidl at Col. 11, lines 3-19 states:

> As an example, suppose the connection is dragged away from the speaker port. As soon as the cursor vacates the snapped region of the port, the connection unlocks and the endpoint once again follows the cursor. Also, the connection is drawn gray again. This is referred to as a "snap-leaving event". An application may even give audio-feedback for snap-enter and snap-leaving events. Now, if the connection was dragged back to the input of the echo unit, the connection would be invalid, since it would result in feedback and the echo unit locking up. On the snap-enter event, the connection's endpoint once again has locked on to an input port. Type-negotiation confirms the right data type, but a topology check

> finds a circular path and feeds back to the user that the connection cannot be made. This feedback stays in place for as long as the cursor remains within the snapping radius of the input port. On the snap-leaving event, the feedback is taken down and the user is free to explore other possibilities.

The above-noted passage in Seidl describes a technique for connecting lines and graphical icons while developing an application program. As described, the technique identifies when a line begins and ends at the same port of an echo-unit icon. *See* Seidl, Figs. 6-7. When the application developer moves her cursor away from the snapping region of the echo unit's input port, the line follows the cursor. However, if the application developer returns the endpoint of the line to the input port, a topology check determines that the line is "circular" (i.e., begins and ends at the same port) and thus cannot be made.

Applicants respectfully submit that Col. 11, lines 3-19 of Seidl is directed to connecting a line to a graphical icon while developing an application program, and therefore fails to disclose or suggest the Applicants' claimed <u>run time</u> busy indicator.

Seidl at Col. 14, lines 17-25 states:

> A test is immediately performed at decision block 1510 to determine if the mouse button is down. If so, then at decision block 1520, a test is performed to determine if tracking snap is active. If so, then at function block 1530, the direction is constrained and at function block 1540, draw feedback is fed back to the application to be displayed to the user. If the mouse button was not down at decision block 1510, then control is passed via function block 1560 to the track end phase which is detailed in Fig. 15.

The above-noted passage in Seidl describes a sequence of steps for tracking a line while developing an application program. *See* Seidl, Col. 2, lines 54-56. Specifically, the line is drawn by the application program developer when her mouse button is depressed (step 1510), and the location of the line on the screen may be constrained at locations of a grid (step 1570) or at snapping regions of graphical icons (step 1520).

Like the other cited passages of Seidl, Col. 14, lines 17-25 is also directed to drawing lines while developing an application program, and therefore fails to disclose or suggest the Applicants' claimed run time busy indicator.

Further, as noted above, the programming aid described in Seidl is not a "busy indicator" as recited in present claim 1. The programming aid described in Seidl relates to preventing an input connection of a program object from feeding back on itself, i.e., feeding back on another input of the same program object. The programming aid described in Seidl has nothing to do with whether the program object is busy executing its function, and is further unrelated to whether the program object is busy executing its function during execution of the flow diagram.

All independent claims are believed to be in condition for allowance.

All dependent claims depend on independent claims which are believed to be in condition for allowance.

Early favorable action is respectfully solicited.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-95500/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☒ Request for Continued Examination

Respectfully submitted,

[illegible]

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 5/27/2005 JCH